**Teste em Linguagem Java e Lógica de Programação**

**- Para Candidatos a Analista de Desenvolvimento Pleno/Sênior -**

**NOME: Alexsandro Pires**

**DATA: 10/01/2021**

**ENUNCIADO GERAL PARA QUESTÕES 1 a 5**

Suponha que você faça parte de uma equipe que desenvolve softwares para diversos clientes.

Um de seus clientes é uma empresa que vende milhares de produtos, dos mais variados tipos e setores: artigos de papelaria, roupas e acessórios, alimentos e bebidas, materiais esportivos, eletrônicos e eletrodomésticos.

Você e sua equipe precisam desenvolver em Java um sistema de controle de produtos: tudo o que esta empresa compra e vende será registrado neste sistema.

A Especificação Técnica do Sistema contém o seguinte requisito de implementação:

|  |
| --- |
| Para Preços de produtos e valores de venda/compra, considerar valores como inteiros, em centavos.  Exemplo: 1,99 -> representado no Sistema como 199 |

Neste cenário, responda às questões 1 a 5 abaixo, considerando “**Sistema**” como o sistema a ser **desenvolvido em Java** por sua Equipe, e “**Empresa**” como o cliente para o qual vocês desenvolverão o sistema.

**Questão 1**:

O Sistema possui um Módulo de Controle de Estoque, que permite realizar vários cálculos. A Equipe do Projeto se reuniu e você ficou responsável pelas tarefas abaixo.

* 1. Declare uma interface chamada Estoque contendo a assinatura de um método calcular que recebe como parâmetro de entrada um inteiro e devolve um inteiro.
  2. Crie uma classe DoisEmUm que implemente Estoque. Na implementação do método calcular, retorne o valor de entrada multiplicado por 2.

*package* br.com.teste.fiserv;  
  
*public interface* Estoque {  
 Integer calcular(Integer param);  
}

*package* br.com.teste.fiserv;  
  
*public class* DoisEmUm *implements* Estoque{  
  
 *@Override  
 public* Integer calcular(Integer param) {  
 *return* param \* 2;  
 }  
}

*package* br.com.teste.fiserv;  
  
*public class* Application {  
 *public static void* main(String[] args) {  
 DoisEmUm doisEmUm = *new* DoisEmUm();  
  
 Integer valorEntrada = 20;  
  
 Integer paramMultiplicado = doisEmUm.calcular(valorEntrada);  
  
 System.out.println("Valor de Entrada: " + valorEntrada + ", Resultado: " + paramMultiplicado);  
 }  
}

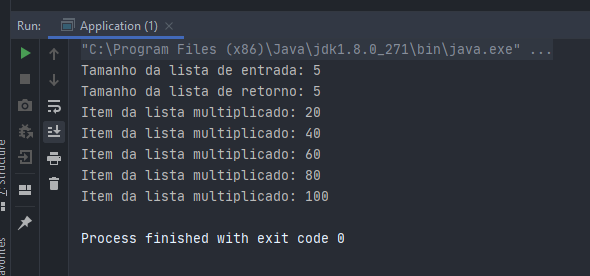
**Questão 2**:

Você realizou os devidos testes no que foi desenvolvido na Questão 1, e verificou que a implementação estava correta. Sua próxima tarefa envolve o uso da classe DoisEmUm criada anteriormente (Questão 1, item b).

Declare a classe Campanha contendo um método calcularPromocao que recebe uma lista de inteiros como parâmetro e devolve uma lista de mesmo tamanho na qual cada elemento é o dobro do valor do elemento correspondente na lista original. A implementação desse método deve criar uma instância de DoisEmUm e usá-la para calcular os valores da lista que será retornada.

*package* br.com.teste.fiserv;  
  
*import* java.util.ArrayList;  
*import* java.util.List;  
*import* java.util.stream.Collectors;  
  
*public class* Campanha {  
  
 *public* List<Integer> calcularPromocao(List<Integer> valores){  
  
 List<Integer> valoresRetorno = *new* ArrayList<>();  
  
 DoisEmUm doisEmUm = *new* DoisEmUm();  
  
 *return* valores.stream().map(doisEmUm::calcular).collect(Collectors.toList());  
 }  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.Arrays;  
*import* java.util.List;  
  
*public class* Application {  
 *public static void* main(String[] args) {  
 Campanha campanha = *new* Campanha();  
  
 List<Integer> valores = Arrays.asList(10, 20, 30, 40, 50);  
 System.out.println("Tamanho da lista de entrada: " + valores.size());  
  
 valores = campanha.calcularPromocao(valores);  
 System.out.println("Tamanho da lista de retorno: " + valores.size());  
  
 valores.forEach(v->System.out.println("Item da lista multiplicado: " + v));  
   
 }  
}



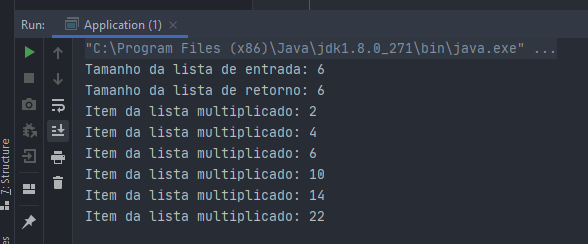
**Questão 3**:

Numa reunião com a Equipe, vocês revisaram o código e verificaram que o método calcularPromocao criado anteriormente (Questão 2) não atendia a uma certa necessidade do Sistema.

1. Reescreva o método calcularPromocao: o método deve receber também Estoque como parâmetro e usar este objeto de entrada em vez de instanciar DoisEmUm.
2. Considere a classe Campanha declarada na Questão 2: ela originalmente continha apenas o método calcularPromocao com apenas um parâmetro de entrada. Atualize esta classe, considerando que ela deve conter também o método calcularPromocao do item a da Questão 3. Faça alterações que julgar necessárias para que a classe possua os dois métodos de cálculo.

*package* br.com.teste.fiserv;  
  
*import* java.util.ArrayList;  
*import* java.util.List;  
*import* java.util.stream.Collectors;  
  
*public class* Campanha {  
  
 *public* List<Integer> calcularPromocao(List<Integer> valores){  
 List<Integer> valoresRetorno = *new* ArrayList<>();  
 DoisEmUm doisEmUm = *new* DoisEmUm();  
 *return* valores.stream().map(doisEmUm::calcular).collect(Collectors.toList());  
 }  
  
 *public* List<Integer> calcularPromocao(List<Integer> valores, Estoque estoque){  
 List<Integer> valoresRetorno = *new* ArrayList<>();  
  
 valores.forEach(v->{valoresRetorno.add(estoque.calcular(v));});  
  
 *return* valoresRetorno;  
 }  
  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.Arrays;  
*import* java.util.List;  
  
*public class* Application {  
 *public static void* main(String[] args) {  
  
 Campanha campanha = *new* Campanha();  
 Estoque estoque = *new* DoisEmUm();  
  
 List<Integer> valores = Arrays.asList(1, 2, 3, 5, 7, 11);  
 System.out.println("Tamanho da lista de entrada: " + valores.size());  
  
 valores = campanha.calcularPromocao(valores, estoque );  
 System.out.println("Tamanho da lista de retorno: " + valores.size());  
  
 valores.forEach(v->System.out.println("Item da lista multiplicado: " + v));  
  
 }  
}



**ENUNCIADO PARA QUESTÕES 4 E 5**

O Setor Administrativo da Empresa solicitou a sua Equipe a inclusão de 4 (quatro) relatórios no Sistema:

1. Relatório Almoxarifado – Total de Canetas em Estoque
2. Relatório Faturamento PROMO - Valor Faturado com Campanhas Promocionais
3. Relatório Faturamento À VISTA- Valor Faturado com Vendas pagas à vista
4. Relatório Faturamento PARCELADO – Valor Faturado com Vendas pagas em até 3 parcelas

A Equipe discutiu a implementação dos quatro relatórios e optou por declarar uma interface Relatorio para atender à solicitação. Esta interface contém a assinatura do método calcularTotal: ele recebe como parâmetro de entrada uma java.util.List de valores e retorna um java.lang.Integer

**Questão 4:**

A Equipe verificou que os Relatórios de Faturamento possuem requisitos similares e você ficou como responsável pelas tarefas abaixo.

1. Criar uma classe abstrata AbstractRelatorioFaturamento que implemente Relatorio: os Relatórios de Faturamento estenderão esta classe.
2. Incluir em AbstractRelatorioFaturamento o método aplicarTaxas, a ser usado por todos os Relatórios de Faturamento. Os parâmetros de entrada do método são um inteiro e uma lista de taxas. O método deve devolver um inteiro, considerando que:
   1. O inteiro recebido como parâmetro de entrada é o valor de uma venda.
   2. Taxa é um valor numérico de 0 a 100.
   3. No dicionário de dados do Sistema, “aplicar uma taxa” significa multiplicar um certo valor pela taxa em questão.
   4. O retorno do método deve ser a aplicação de todas as taxas ao valor da venda.

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public interface* Relatorio {  
  
 Integer calcularTotal(List<Integer> valores, Integer valor);  
}

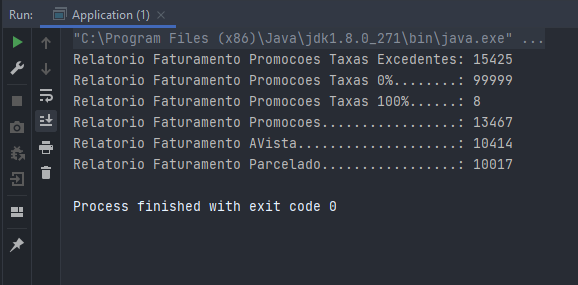
*package* br.com.teste.fiserv;  
  
*import* java.math.BigDecimal;  
*import* java.util.List;  
  
*abstract class* AbstractRelatorioFaturamento *implements* Relatorio {  
  
 *public* Integer aplicarTaxas(Integer valorEntrada, List<Integer> taxas) {  
  
  
 BigDecimal valorAcumulado = converterInteiroParaBigDecimal(valorEntrada);  
  
 *for* (Integer taxa : taxas) {  
  
 BigDecimal valorEntradaBigDecimal = converterInteiroParaBigDecimal(valorEntrada);  
 BigDecimal taxaBigDecimal = converterInteiroParaBigDecimal(taxa);  
 taxaBigDecimal = taxaBigDecimal.divide(*new* BigDecimal("100"));  
  
 *if* ((taxaBigDecimal.compareTo(BigDecimal.ZERO) == 1) || (taxaBigDecimal.compareTo(BigDecimal.ZERO) == 0)) {  
 *if* ((taxaBigDecimal.compareTo(*new* BigDecimal("100")) == -1) || (taxaBigDecimal.compareTo(*new* BigDecimal("100")) == 0)) {  
  
 BigDecimal valorTaxa = valorEntradaBigDecimal.multiply(taxaBigDecimal);  
  
 valorAcumulado = valorAcumulado.add(valorTaxa).setScale(2, BigDecimal.ROUND\_HALF\_UP);  
 }  
 }  
 }  
 *return* converterBigDecimalParaInteiro(valorAcumulado);  
 }  
  
 *private* BigDecimal converterInteiroParaBigDecimal (Integer valorInteiro){  
 BigDecimal valorBigDecimal = BigDecimal.valueOf(valorInteiro);  
 *return* valorBigDecimal.movePointLeft(2);  
 }  
  
 *private* Integer converterBigDecimalParaInteiro (BigDecimal valorBigDecimal){  
 valorBigDecimal = valorBigDecimal.movePointRight(2);  
 *return* valorBigDecimal.intValueExact();  
 }  
  
 }

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioFaturamentoAVista *extends* AbstractRelatorioFaturamento *implements* Relatorio{  
  
 *@Override  
 public* Integer calcularTotal(List<Integer> taxas, Integer valorEntrada) {  
 *return* aplicarTaxas(valorEntrada, taxas);  
 }  
  
 *@Override  
 public* Integer aplicarTaxas(Integer valorEntrada, List<Integer> taxas) {  
 *return super*.aplicarTaxas(valorEntrada, taxas);  
 }  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioFaturamentoParcelado *extends* AbstractRelatorioFaturamento{  
  
 *@Override  
 public* Integer calcularTotal(List<Integer> taxas, Integer valorEntrada) {  
 *return* aplicarTaxas(valorEntrada, taxas);  
 }  
  
 *@Override  
 public* Integer aplicarTaxas(Integer valorEntrada, List<Integer> taxas) {  
 *return super*.aplicarTaxas(valorEntrada, taxas);  
 }  
  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioFaturamentoPromo *extends* AbstractRelatorioFaturamento{  
  
 *@Override  
 public* Integer calcularTotal(List<Integer> taxas, Integer valorEntrada) {  
 *return* aplicarTaxas(valorEntrada, taxas);  
 }  
  
 *@Override  
 public* Integer aplicarTaxas(Integer valorEntrada, List<Integer> taxas) {  
 *return super*.aplicarTaxas(valorEntrada, taxas);  
 }  
  
}

*package* br.com.teste.fiserv;  
  
*import* java.math.BigDecimal;  
*import* java.util.Arrays;  
*import* java.util.List;  
  
*public class* Application {  
 *public static void* main(String[] args) {  
  
 List<Integer> taxas = Arrays.asList(10199, -38, 0);  
 RelatorioFaturamentoPromo promoExcedente = *new* RelatorioFaturamentoPromo();  
 System.out.println("Relatorio Faturamento Promocoes Taxas Excedentes: " + promoExcedente.calcularTotal(taxas, 15425));  
  
 taxas = Arrays.asList(0);  
 RelatorioFaturamentoPromo promozeroPorCento = *new* RelatorioFaturamentoPromo();  
 System.out.println("Relatorio Faturamento Promocoes Taxas 0%........: " + promoExcedente.calcularTotal(taxas, 99999));  
  
 taxas = Arrays.asList(10000);  
 RelatorioFaturamentoPromo promoCemPorCento = *new* RelatorioFaturamentoPromo();  
 System.out.println("Relatorio Faturamento Promocoes Taxas 100%......: " + promoExcedente.calcularTotal(taxas, 4));  
  
 taxas = Arrays.asList(199, -1, 38, 10001, 6);  
 RelatorioFaturamentoPromo promo = *new* RelatorioFaturamentoPromo();  
 System.out.println("Relatorio Faturamento Promocoes.................: " + promo.calcularTotal(taxas, 13147));  
  
 taxas = Arrays.asList(213, 25, 9, 167);  
 RelatorioFaturamentoAVista aVista = *new* RelatorioFaturamentoAVista();  
 System.out.println("Relatorio Faturamento AVista....................: " + aVista.calcularTotal(taxas, 10000));  
  
 taxas = Arrays.asList(17, -50);  
 RelatorioFaturamentoParcelado parcelado = *new* RelatorioFaturamentoParcelado();  
 System.out.println("Relatorio Faturamento Parcelado.................: " + parcelado.calcularTotal(taxas, 10000));  
  
 }  
}



**Questão 5:**

A Equipe desenvolveu os quatros Relatórios e você ficou encarregado pela revisão de código da classe RelatorioCanetasEmEstoque.

Esta classe implementa o “Relatório Almoxarifado: Total de Canetas em Estoque”.

Considere os seguintes requisitos de negócio e de implementação:

* + No estoque, as canetas são estocadas em caixas fechadas e cada caixa contém 50 canetas.
  + No dicionário de dados do Sistema, um “Valor de Evento de Estoque” é a quantidade de itens que entram ou saem do Estoque. Por convenção, valor negativo indica saída do Estoque.
  + O Sistema registra todos os valores de evento de Estoque.

Considere o código abaixo que implementa o método calcularTotal em RelatorioCanetasEmEstoque, sendo que, na chamada do método, sabemos que:

* o parâmetro de entrada listaValores é um Objeto java.util.ArrayList;
* cada elemento desta lista é um valor de evento de estoque.

Este código gera uma falha no Relatório: ele não contém defeitos de sintaxe, mas não apresenta o resultado esperado.

1. Indique a linha onde é produzido o erro e como corrigí-lo**.**

**Resposta:** O erro esta na linha 23

O problema so acontece quando na listaValores temos um numero negativo, pois a validação em questão esta não cumpre a regra de sinais para a situação de negocio, em operações em que temos sinais iguais a soma prevalesce.

Para resolver o problema e so inverter a operação em questão, logo, onde era subtração passa a ser soma.

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioAlmoxarifado *implements* Relatorio{  
  
 *@Override  
 public* Integer calcularTotal(List<Integer> valores, Integer valor) {  
 *return null*;  
 }  
  
 *public* Integer calcularTotal(List<Integer> quantidadeEntradaOuSaidaDeCanetas) {  
 RelatorioCanetasEmEstoque emEstoque = *new* RelatorioCanetasEmEstoque();  
 *return* emEstoque.calcularTotal(quantidadeEntradaOuSaidaDeCanetas);  
 }  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioAlmoxarifado *implements* Relatorio{  
  
 *@Override  
 public* Integer calcularTotal(List<Integer> valores, Integer valor) {  
 *return null*;  
 }  
  
 *public* Integer calcularTotal(List<Integer> quantidadeEntradaOuSaidaDeCanetas) {  
 RelatorioCanetasEmEstoque emEstoque = *new* RelatorioCanetasEmEstoque();  
 *return* emEstoque.calcularTotal(quantidadeEntradaOuSaidaDeCanetas);  
 }  
}

*package* br.com.teste.fiserv;  
  
*import* java.math.BigDecimal;  
*import* java.util.ArrayList;  
*import* java.util.Arrays;  
*import* java.util.List;  
  
*public class* Application {  
 *public static void* main(String[] args) {  
  
 RelatorioAlmoxarifado relatorioAlmoxarifado = *new* RelatorioAlmoxarifado();  
 Integer resultado = relatorioAlmoxarifado.calcularTotal(Arrays.asList(5, 3, 4, -1));  
***// 5 + 3 + 4 + (-1) = 11 \* 50 = 550***System.out.println("Total de Canetas no Estoque: " + resultado);  
 }  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioCanetasEmEstoque {  
  
 *public* Integer calcularTotal(List listaValores) {  
  
 Integer result;  
 *int* taxa = 0;  
 *int* contador;  
 *int* aux = -1;  
  
 *if* (listaValores == *null*) {  
 result = *null*;  
 } *else* {  
  
 *int* total = 0;  
 *int*[] valoresEmInteiros = *new int*[listaValores.size()];  
 *int* tamanho = valoresEmInteiros.length;  
  
 *for* (contador = 1; contador <= tamanho; contador++) {  
 *int* indice = contador - 1;  
 Integer temp = (Integer) listaValores.get(indice);  
 valoresEmInteiros[indice] = temp.intValue();  
 }  
  
 *for* (contador = 1; contador <= tamanho; contador++) {  
 *int* indice = contador - 1;  
 aux = valoresEmInteiros[indice];  
  
 *if* (aux > 0) {  
 total = total + aux;  
 } *else if* (aux < 0) {  
 **total = total + aux;**   
 } *else* {  
 total = total + valoresEmInteiros[indice];  
 }  
  
 taxa = taxa \* aux;  
 aux = 50;  
 valoresEmInteiros[indice] = '\0';  
 }  
  
 result = *new* Integer(total \* aux);  
 }  
 *return* result;  
 }  
  
}

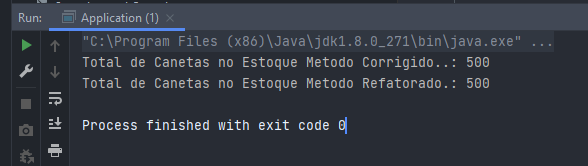
1. Explique, com suas próprias palavras, que melhorias você faria no código abaixo.

**Resposta:** Criaria um método que recebesse uma lista e para tal, validaria se o item da lista não e nulo e multiplicaria por 50 e incrementaria o resultado em um atributo “totalDeCanetasNoEstoque”, que ao fim do looping retorna com a quantidade de canetas no estoque, lenvando em consideração entrada e saída.

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
  
*public class* RelatorioAlmoxarifado *implements* Relatorio{  
  
 *@Override  
 public* Integer calcularTotal(List<Integer> valores, Integer valor) {  
 *return null*;  
 }  
  
 *public* Integer calcularTotal(List<Integer> listaQuantidadeDeCaixasEntradaOuSaida) {  
 RelatorioCanetasEmEstoque emEstoque = *new* RelatorioCanetasEmEstoque();  
 *return* emEstoque.calcularTotal(listaQuantidadeDeCaixasEntradaOuSaida);  
 }  
  
 *public* Integer calcularTotalRefatorado(List<Integer> listaQuantidadeDeCaixasEntradaOuSaida) {  
 RelatorioCanetasEmEstoque emEstoque = *new* RelatorioCanetasEmEstoque();  
 *return* emEstoque.calcularTotalRefatorado(listaQuantidadeDeCaixasEntradaOuSaida);  
 }  
}

*package* br.com.teste.fiserv;  
  
*import* java.util.List;  
*import* java.util.Objects;  
  
*public class* RelatorioCanetasEmEstoque {  
  
 *public* Integer calcularTotalRefatorado(List<Integer> listaQuantidadeDeCaixasEntradaOuSaida) {  
 *final* Integer QUANTIDADE\_DE\_CANETAS\_POR\_CAIXA = 50;  
 Integer totalDeCanetasNoEstoque = 0;  
  
 *for* (Integer quantasDeCaixaNoEvento : listaQuantidadeDeCaixasEntradaOuSaida) {  
 *if* (Objects.nonNull(quantasDeCaixaNoEvento)) {  
 totalDeCanetasNoEstoque = totalDeCanetasNoEstoque + quantasDeCaixaNoEvento \* QUANTIDADE\_DE\_CANETAS\_POR\_CAIXA;  
 }  
  
 }  
 *return* totalDeCanetasNoEstoque;  
 }

*package* br.com.teste.fiserv;  
  
*import* java.math.BigDecimal;  
*import* java.util.ArrayList;  
*import* java.util.Arrays;  
*import* java.util.List;  
  
*public class* Application {  
 *public static void* main(String[] args) {  
  
 RelatorioAlmoxarifado relatorioAlmoxarifado = *new* RelatorioAlmoxarifado();  
 Integer resultado = relatorioAlmoxarifado.calcularTotal(Arrays.asList(-1, 5, 3, 4, -1));  
*// 5 + 3 + 4 + (-1) = 11 \* 50 = 550* System.out.println("Total de Canetas no Estoque Metodo Corrigido..: " + resultado);  
  
 resultado = relatorioAlmoxarifado.calcularTotalRefatorado(Arrays.asList(-1, 5, 3, 4, -1));  
*// 5 + 3 + 4 + (-1) = 11 \* 50 = 550* System.out.println("Total de Canetas no Estoque Metodo Refatorado.: " + resultado);  
  
 }  
}



|  |  |
| --- | --- |
| 1 | public Integer calcularTotal(List listaValores) { |
| 2 | Integer result; |
| 3 | int taxa = 0; |
| 4 | int contador; |
| 5 | int aux = -1; |
| 6 | if (listaValores == null) { |
| 7 | result = null; |
| 8 | } else { |
| 9 | int total = 0; |
| 10 | int[] valoresEmInteiros = new int[listaValores.size()]; |
| 11 | int tamanho = valoresEmInteiros.length; |
| 12 | for (contador = 1; contador <= tamanho ; contador++) { |
| 13 | int indice = contador - 1; |
| 14 | Integer temp = (Integer) listaValores.get(indice) ; |
| 15 | valoresEmInteiros[indice] = temp.intValue(); |
| 16 | } |
| 17 | for (contador = 1; contador <= tamanho; contador++) { |
| 18 | int indice = contador - 1; |
| 19 | aux = valoresEmInteiros[indice]; |
| 20 | if (aux > 0 ) { |
| 21 | total = total + aux; |
| 22 | } else if (aux < 0) { |
| 23 | total = total - aux; <- Errado  total = total **+** aux; <- Correto |
| 24 | } else { |
| 25 | total = total + valoresEmInteiros[indice]; |
| 26 | } |
| 27 | taxa = taxa \* aux; |
| 28 | aux = 50; |
| 29 | valoresEmInteiros[indice] = '\0'; |
| 30 | } |
| 31 | result = new Integer(total \* aux); |
| 32 | } |
| 33 | return result; |
| 34 | } |